Ink Services

Understanding the New Ink API

Session 428

Ink Services

Understanding the New Ink API

Giulia Pagallo

Manager, Text Input Group

Handwriting Recognition on Jaguar...

The requests…

Today…

- Introduction
- Technology
- Ink API
- Developer Scenarios

Device Requirements

- Jaguar
  - Wacom tablet and driver
- Panther
  - Wacom tablet and driver
- Devices that generate x, y, and pressure information
  - Real time
  - Off-Line

5 Ink Concepts

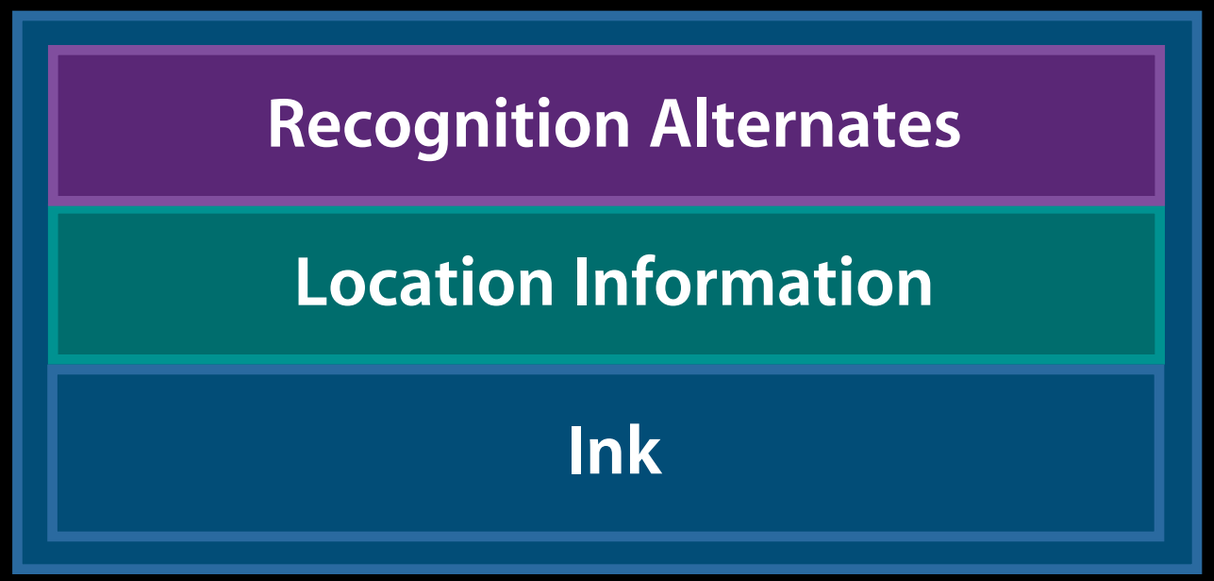
- Ink Phrase
- Phrase Termination
- InkTextRef
- Alternates Menu
- Gestures

Ink Phrase

Phrase Termination

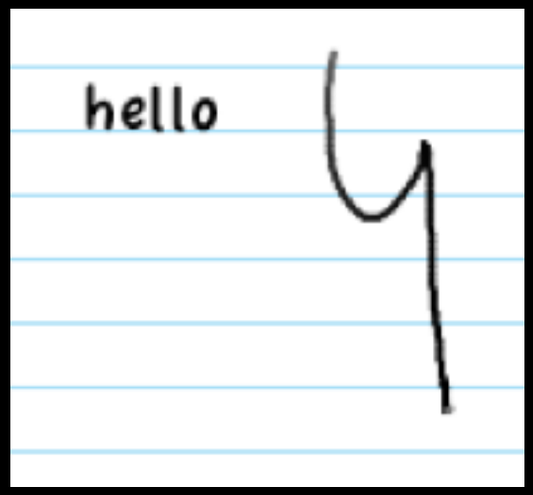
- By default occurs when
  - Stylus goes out of proximity
  - Enough time has elapsed
  - There is a large separation between strokes

InkTextRef

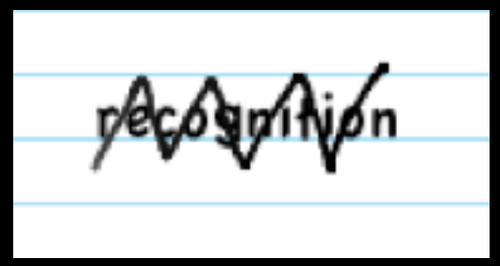


Alternates Menu

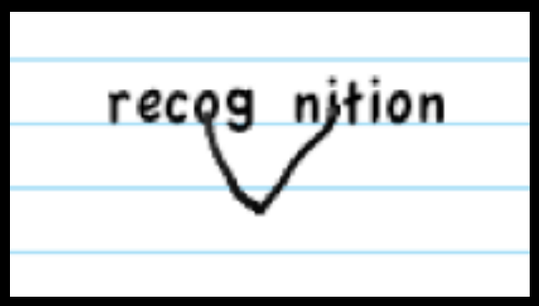
Gestures



Non-targeted
Undo



Optionally Targeted
Erase



Always Targeted
Join

Ink Components

Ink Input Method

- Data collection
- Ink rendering

Ink Framework

- Ink API

Ink Server

- Ink segmentation
- Recognition
- Language model
- Ink window

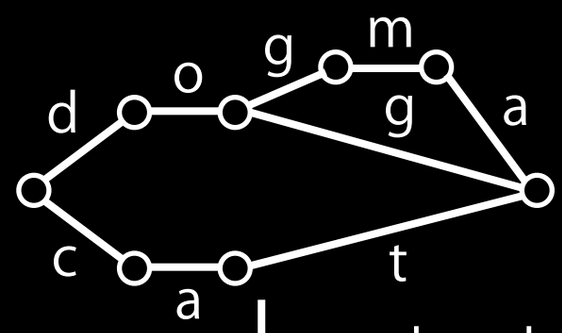
User Preferences

- User settings
- Recognition mode

Ink Technologies

Larry Yaeger
Distinguished Scientist

Recognition Technology



(x,y) points & pen-lifts

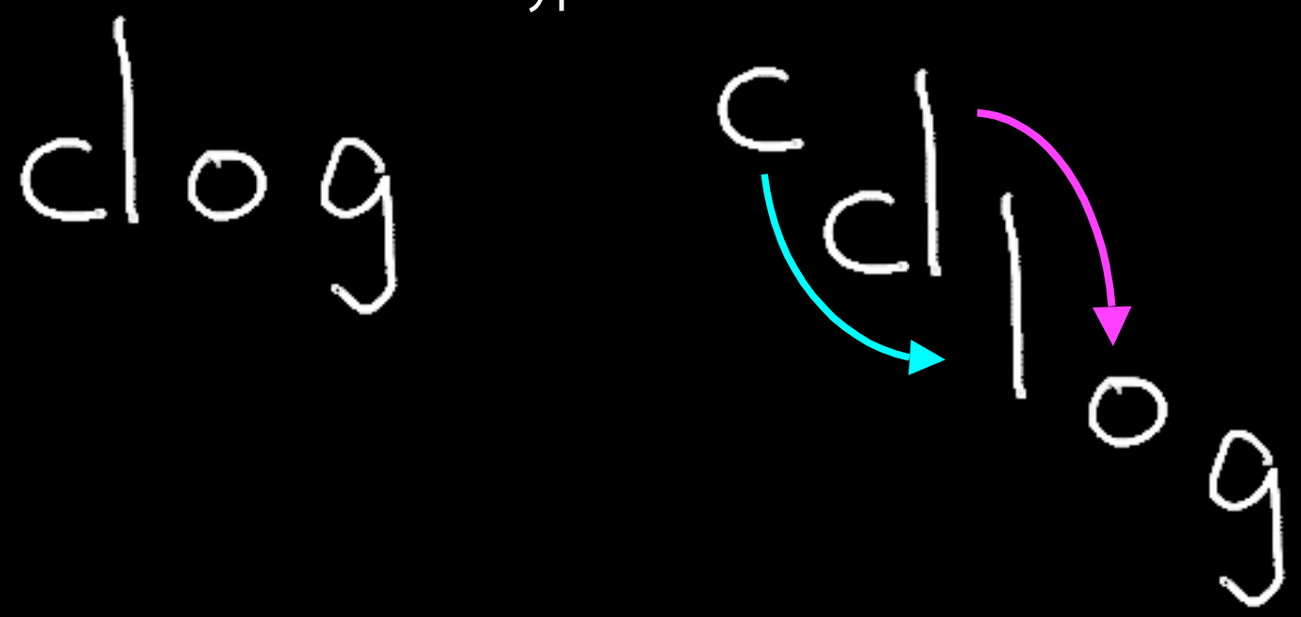
word probabilities

Tentative Segmentation

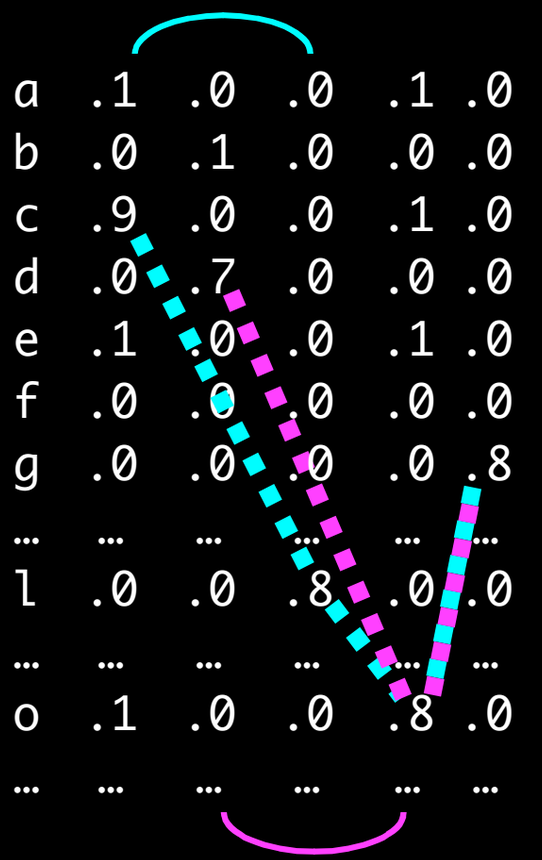
Neural Network Classifier

Beam Search with Context

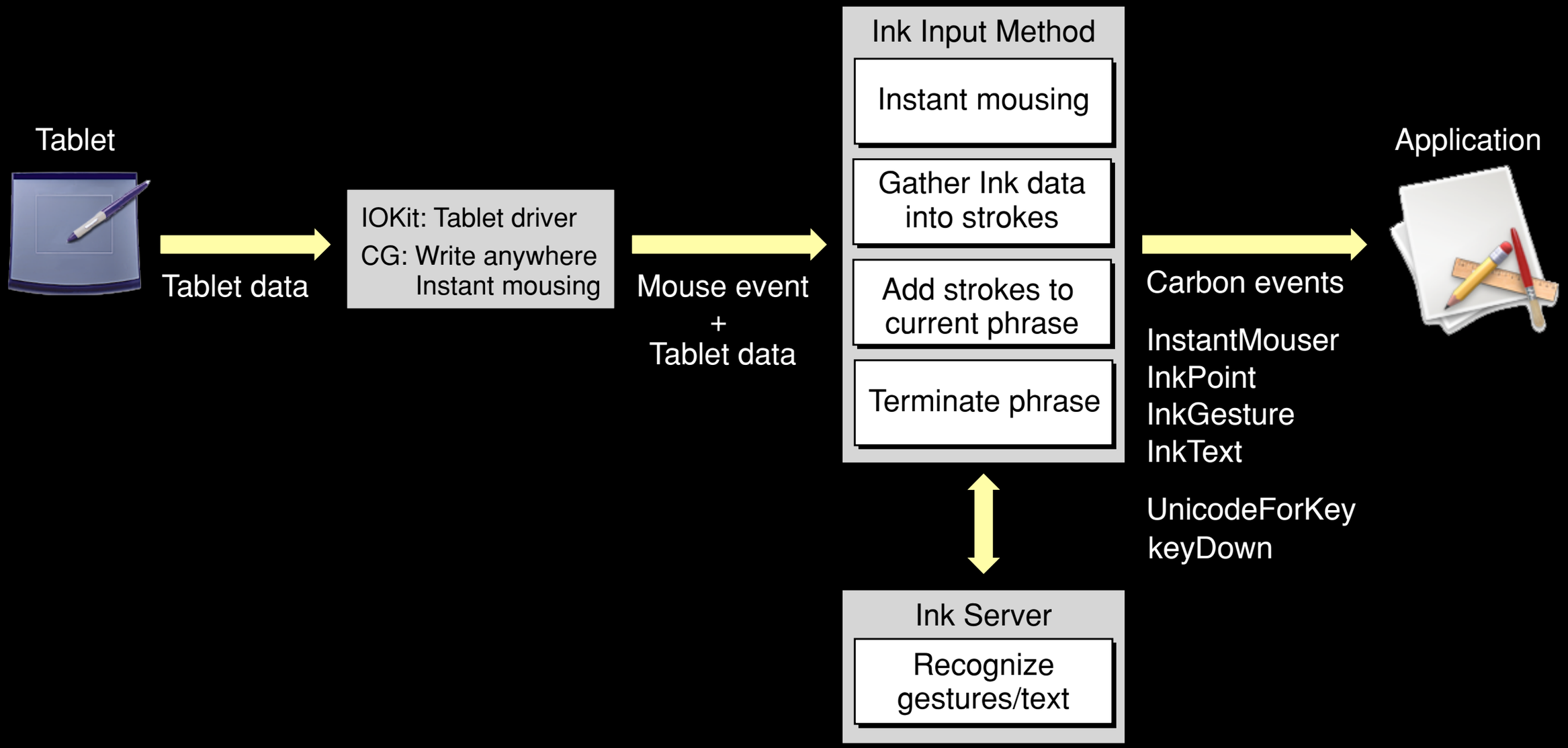
character segmentation hypotheses



character class hypotheses



Ink Data Flow



Ink API

Quick Reference

Ink Events

{ kEventClassApplication, kEventAppIsEventInInstantMouser }

{ kEventClassInk, kEventInkPoint }
{ kEventClassInk, kEventInkGesture }
{ kEventClassInk, kEventInkText }

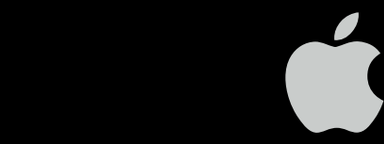
Quick Reference

Ink Objects

InkTextRef

Ink Text API

InkTextAlternatesCount()
InkTextCreateCFString()
InkTextInsertAlternatesInMenu()
InkTextKeyModifiers()
InkTextCopy()
InkTextBounds()
InkTextDraw()
InkTextFlatten()
InkTextCreateFromCFData()



Quick Reference

Ink State

InkUserWritingMode()
InkSetApplicationWritingMode()

InkSetApplicationRecognitionMode()
InkSetPhraseTerminationMode()
InkSetDrawingMode()
InkIsPhraseInProgress()

Quick Reference

Ink Data Types

```
struct InkPoint {
    HIPoint             point;
    TabletPointRec      tabletPointData;
    UInt32              keyModifiers;
};
```

Ink Phrase

```
InkAddStrokeToCurrentPhrase()
InkTerminateCurrentPhrase()
```

Developer Scenarios

Brad Reigel
Senior Software Engineer

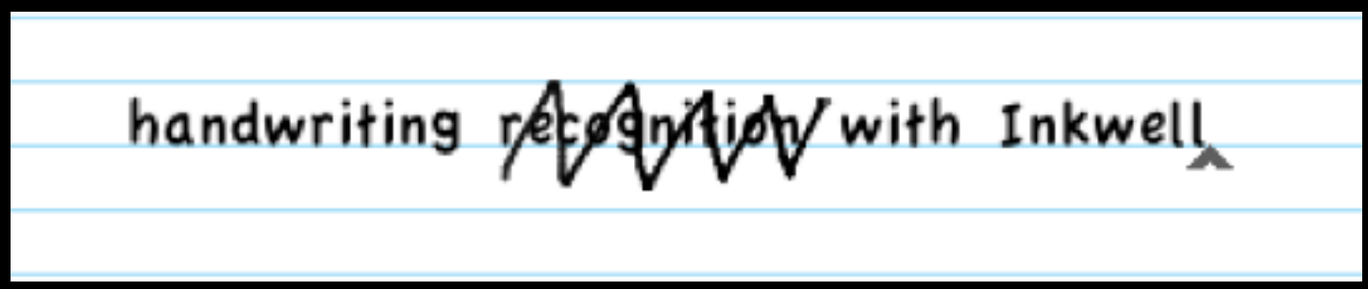
Developer Scenarios

What we're going to cover…

- Text Engines
- Input Devices
- Controlling Phrase Termination

Scenario 1—Text Engines

Interacting with Gestures

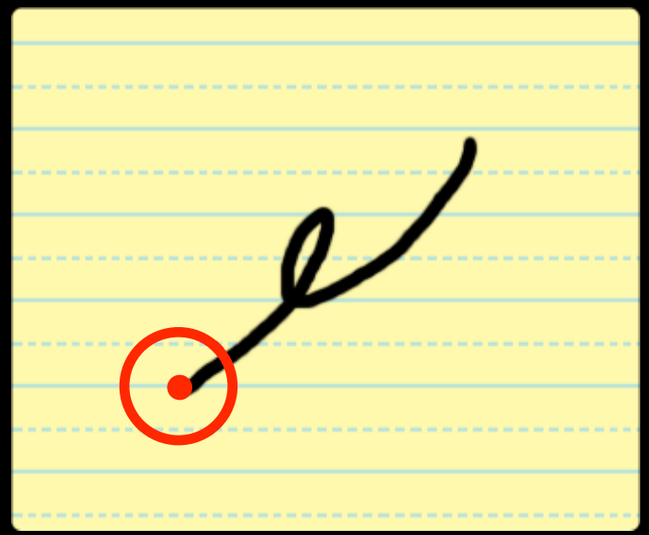


Gesture Events

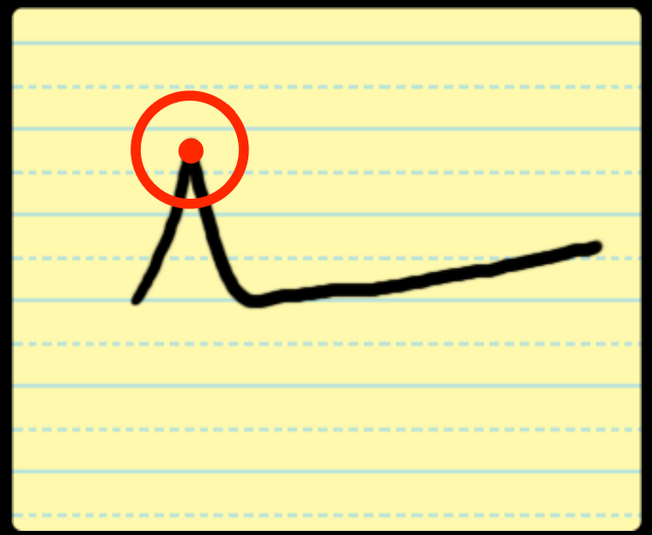
```
kEventClassInk / kEventInkGesture

 -> kEventParamInkGestureKind          // Kind
 -> kEventParamInkGestureBounds        // Bounds
 -> kEventParamInkGestureHotspot       // Hotspot
```

Hotspots and Bounds

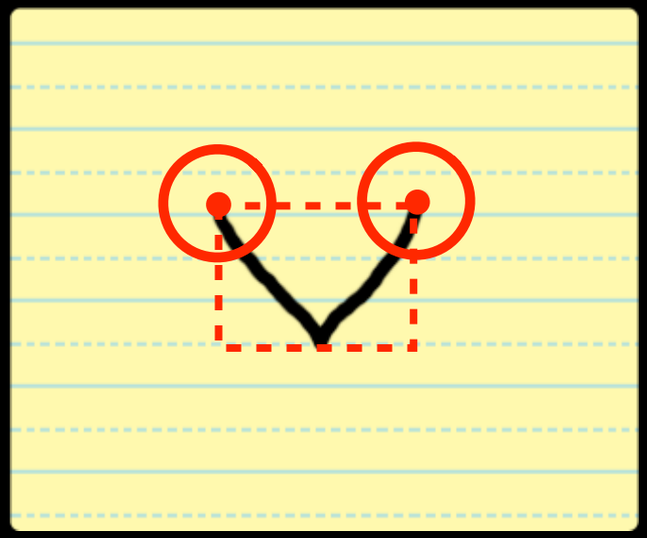
Determining gesture target



kInkGestureCut



kInkGestureRightSpace



kInkGestureJoin

Gesture Event Handler

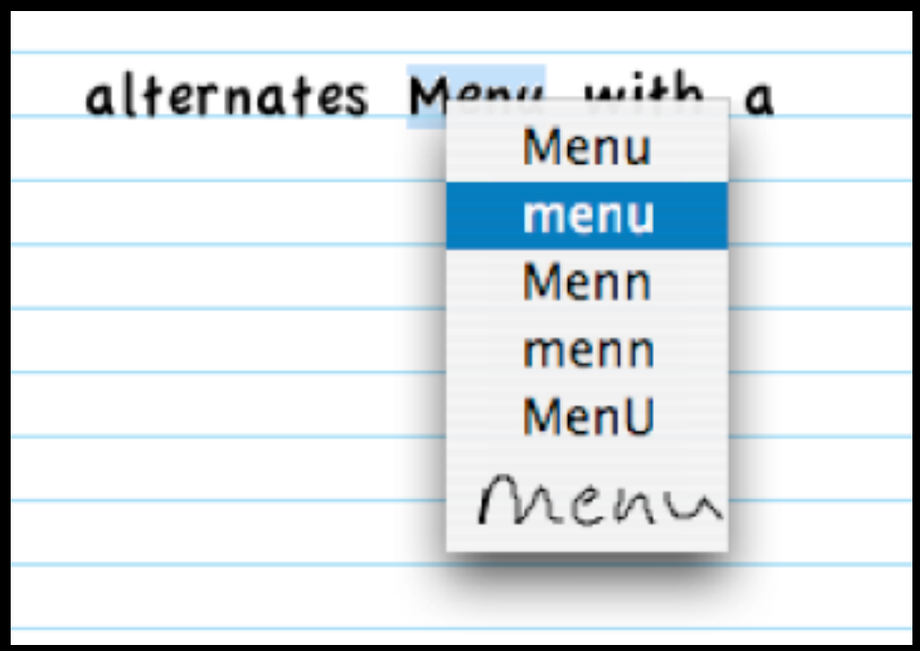
```
{

   UInt32     myGestureKind;
   HIRect     myHotspot;
   HIPoint    myBounds;

  GetEventParameter( eventRef, kEventParamInkGestureKind, typeUInt32,
                     NULL, sizeof( UInt32 ), NULL, &myGestureKind );

  GetEventParameter( eventRef, kEventParamInkGestureHotspot, typeHIPoint,
                     NULL, sizeof( HIPoint ), NULL, &myHotspot );

  GetEventParameter( eventRef, kEventParamInkGestureBounds, typeHIRect,
                     NULL, sizeof( HIRect ), NULL, &myBounds );

  // Continued...
```

Gesture Event Handler (Cont.)

```
switch( myGestureKind ) {

  case kInkGestureUndo:
    MyDoUndoGesture();
    result = noErr;
    break;

  case kInkGestureCut:
    MyDoCutGesture( myHotspot );
    result = noErr;
    break;

  case kInkGestureJoin:
    if( MyDoJoinGesture( myBounds ) == true )
      result = noErr;                  // We handled it
    else
      result = eventNotHandledErr;     // It didn't fall on text
    break;
  }
}
```

Scenario 2—Text Engines

Alternates menu



InkText Events

```
kEventClassInk / kEventInkText

 ->  kEventParamInkTextRef      // Contains InkTextRef
```

InkText Event Handler

```
{
   InkTextRef   myInkTextRef;
   CFStringRef  myCFStringRef;
   Boolean      myIsKeyboardShortcut;

   // Is this a keyboard shortcut?
   GetEventParameter( myInEvent, kEventParamInkKeyboardShortcut, typeBoolean,
                      NULL, sizeof(Boolean), NULL, &myIsKeyboardShortcut );

   if( myIsKeyboardShortcut == false )
   {
      // Extract InkTextRef from event
      GetEventParameter( myInEvent, kEventParamInkTextRef, typePtr,
                         NULL, sizeof(Ptr), NULL, &myInkTextRef );

      // Ask for the top-choice text
      myCFStringRef = InkTextCreateCFString( myInkTextRef, 0 );

      // Insert text - and associated InkTextRef! - into document
      MyAddCFStringToDocument( myCFStringRef, myInkTextRef );
   }
}
```

Implementing an Alternates Menu

```
{

   MenuRef       myMenuRef = NewMenu(...);
   ItemCount     myMenuItems;
   long          mySelection, myMenuItem;
   CFStringRef   myCFString;

   myMenuItems = InkInsertAlternatesInMenu( myInkTextRef, myMenuRef, 0 );

   // Show Menu and Handle Menu Selection
   selection = PopUpMenuSelect( myMenuRef, location.v, location.h, 1 );

   // Get the menuItem
   myMenuItem = selection & 0x00ff;

   // Get the text for that menu item
   CopyMenuItemTextAsCFString( myMenuRef, myMenuItem, &myCFString );

   MyChangeCurrentWordInDocument( myCFString );

}
```

Scenario 3—Input Devices

Handwriting recognition solutions

- Collect Ink data
- Perform recognition

Collecting Data and Submitting It for Recognition

- Set recognition state with InkSetApplicationWritingMode()
- Gather and store tablet data from mouse down and dragged events
  - Call InkAddStrokeToCurrentPhrase() for each stroke
  - Call InkTerminateCurrentPhrase() when finished adding strokes
  - Store data into arrays of InkPoints (strokes)
  - One down-dragged-up sequence is a stroke

Initial Setup

```
{

  // Install mouse event handler
  MyInstallMouseEventHandler();

  // Turn off automatic recognition in this application
  InkSetApplicationWritingMode( kInkWriteNowhereInApp );

}
```

Mouse Event Handler

```
{

  InkPoint *myInkPointArray;

  myEventKind = GetEventKind( myMouseEventRef );

  if( myEventKind == kEventMouseDown )
  {
    myInkPointArray = MyStartNewArray( myMouseEventRef );
    myInkPointCount = 1;
  }
  else if( myEventKind == kEventMouseDragged )
  {
    MyAddPointToArray( myMouseEventRef, myInkPointArray );
    myInkPointCount++;
  }
  else if( myEventKind == kEventMouseUp )
    InkAddStrokeToCurrentPhrase( myInkPointCount, myInkPointArray );

}
```

Finished Gathering Data

```
{

  // When finished accumulating data...
  InkTerminateCurrentPhrase();

}
```

Scenario 4—
Controlling Phrase Termination

- Ink Point events
- Phrase termination

Controlling Termination

- Automatic termination can be turned off
  - InkSetPhraseTermination()
- Application can terminate phrase
  - InkTerminateCurrentPhrase()

InkPoint Events

One More Event…

```
kEventClassInk / kEventInkPoint    // Sent for each mouse event

 ->  kEventParamEventRef           // Contains mouse event
```

- Sent for each mouse down and dragged event that Ink handles during a phrase
- Returning noErr causes Ink to ignore that mouse event
- Examine the enclosed mouse event to determine location

Custom Termination

- Call InkSetPhraseTermination() to turn off automatic termination
- InkPoint event handler checks location of mouseDown events
  - If mouseDown lands on “Terminate Phrase” button, and if InkIsInking() returns true, then
    - Call InkTerminateCurrentPhrase() to terminate the phrase
  - Else return eventNotHandledErr

Setup

```
{

   EventTypeSpec kMyInkPointEvent = { kEventClassInk, kEventInkText };

   InstallEventHandler( GetApplicationEventTarget(), MyInkPointHandler,
      GetEventTypeCount( kMyInkPointEvent ), &kMyInkPointEvent, window, NULL );

   InkSetPhraseTermination( kInkTerminationNone );

}
```

InkPoint Event Handler

```
{

   // Extract mouse EventRef from InkPoint event
   GetEventParameter( myInkPointEventRef, kEventParamEventRef, typeEventRef,
                      NULL, sizeof(EventRef), NULL, &myMouseEventRef );

   // Check if this was a mouseDown, in the button, while inking
   if( GetEventKind( myMouseEventRef ) == kEventMouseDown )
   {
      if( MyTestForButtonHit( myMouseEventRef ) == true )
      {
         if( InkIsInking() == true )
         {
            InkTerminateCurrentPhrase();
            return( noErr );
         }
      }
   }
   return( eventNotHandledErr );

}
```

Remember

- Less than 20 API
- Leverage Carbon event model
- Go beyond the built-in support…

Ink Different!

For More Information

Reference Library

Documentation

Inside Mac OS X: Ink Services Reference (WWDC-only distribution)

ADC Member Site > Download Software >

Inside Mac OS X: Using Ink Services in Your Application (WWDC-only distribution)

ADC Member Site > Download Software >

Reference Library

Sample Code

InkSample
(WWDC-only distribution)
ADC Home > Sample Code >

Other Resources

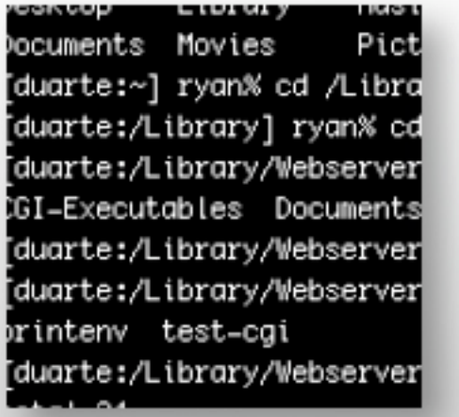
Recognition Technology
http://pobox.com/~larryy

Who to Contact

Xavier Legros
Mac OS X Evangelist
xavier@apple.com

http://developer.apple.com/wwdc2003/urls.html

Q&A



http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference